Student: Dulat Turganbek

Id:26301
Topic is: "Yandex taxi"

I chose a online taxi ordering system like yandex taxi purpose of this system is provide convenience to users of this system, the main task of this system is to order a taxi using this system, customers can order a taxi from point A to point B for this, customers write data about themselves, this means their name, phone number and write their location and place where they want to go, there are types of taxis such as economy class or premium class, there are also types of payment such as cash or payment by card and etc.

Design Patterns

Creational Patterns

1)Builder pattern I used this pattern to create a customer (orderer) using this pattern we add information about the customer such as name, phone number, point A (place of ordering a taxi or location of the client), point B (place where the client wants to go).

2)Singleton pattern the project has a Singleton class in this class there is a method I used this pattern to run this method in the Singleton class and for this process this pattern was used.

Structural Patterns

3)Facade pattern this pattern was used to facilitate the client application, I opened the facade class and wrote all the methods there and called the facade in the main class.

4)Proxy pattern this pattern was used to make a card or cash payment method.

Behavioral Patterns

5)Chain of Responsibility

Using this pattern, I filled in the driver's data, first for each information I created a separate class, such as the driver's number, driver's name, driver's car number, etc., first I created an abstract class with a method and created secondary classes that extends the method of the abstraction class.

6)State Pattern

State pattern this pattern was used to show states (path) of order like new order, or taxi on the way or taxi driver arrived at the destination.

Payment

Interface: Payment
- pay(int) void

Class: Cash (implements Payment)
- pay(int) void

Class: Card (implements Payment)
- pay(int) void

Singleton

Class: Singleton
- yandexTaxi : Singleton
- Singleton()
- getYandexTaxi() Singleton
- print() void

Main

Class: Main
- main(String[]) void

States

Interface: States
- NewOrder(Order) void
- OrderConfirmed(Order) void
- Arrived(Order) void

Class: Confirmed (implements States)
- Confirmed()
- NewOrder(Order) void
- OrderConfirmed(Order) void
- Arrived(Order) void

Class: NewOrderr (implements States)
- NewOrderr()
- NewOrder(Order) void
- OrderConfirmed(Order) void
- Arrived(Order) void

Class: Arrived (implements States)
- Arrived()
- NewOrder(Order) void
- OrderConfirmed(Order) void
- Arrived(Order) void

Class: Order
- Order(States)
- NewOrder() void
- OrderConfirmed() void
- Arrived() void
- currentState : States